GraSP: A Matlab Toolbox for Graph Signal Processing

Benjamin Girault[⋆◇], **Shrikanth S. Narayanan**[◇], **Antonio Ortega**[◇], **Paulo Gonçalves**[⋆], **Éric Fleury**[⋆]

[⋆] Université de Lyon, Inria, ENS de Lyon, CNRS, UCB Lyon 1, 69342, Lyon, FRANCE

[◇] Signal and Image Processing Institute, University of Southern California, CA 90089 Los Angeles, USA

benjamin.girault@usc.edu, {shri, ortega}@sipi.usc.edu, {paulo.goncalves, eric.fleury}@ens-lyon.fr

Objectives

- Matlab toolbox to create, process and analyze graph signals.
- Framework for the community to easily share and experiment.

Initialization

```
% Install the toolbox (one time operation)
grasp_install;

% Start the toolbox (every time Matlab is started)
grasp_start;
```

Figure 1: *GraSP initialization.*

Plotting Graphs and Graph Signals

```
% Build a graph and a graph signal
g   = grasp_plane_rnd(100);
g.A = grasp_adjacency_thresh(g, 0.01);

% Compute the graph Fourier transform
g   = grasp_eigendecomposition(g);

% Some signals
x_1 = grasp_heat_kernel(g, 2);              % Heat kernel
x_2 = normrnd(0, 1, grasp_nb_nodes(g), 1); % White noise

% Display one of them
grasp_show_graph(gca, g, 'node_values', x_1);
```

Figure 2: *Matlab code to generate Figure 3.*

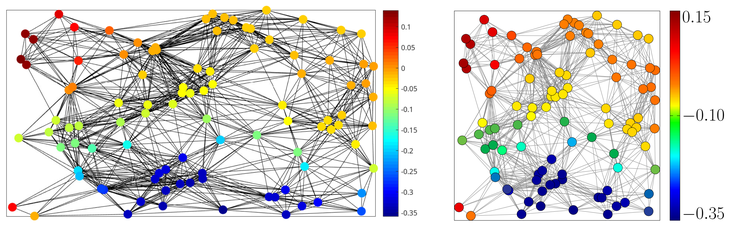

Figure 3: *A random sensor network depicted using Matlab (left) and LaTeX(right).*

```
% Perform a low pass operation on x
y_hat = grasp_fourier(g, x_2);
y_hat(g.eigvals >= 0.5) = 0;
y = grasp_fourier_inverse(g, y_hat);

% Build a low pass filter
H = grasp_fourier_inverse(g, diag(g.eigvals < 0.5));
y = H * x_2;
```

Figure 4: *Ideal low-pass filter.*

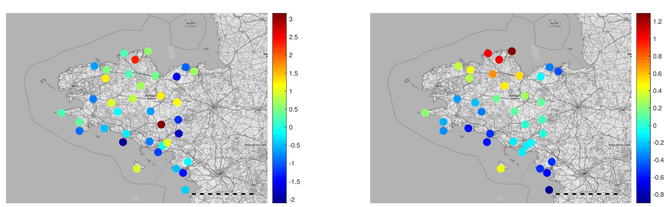

Figure 5: *A weather station graph with white noise (left) and low-pass filtered WN (right).*

Interfacing with LaTeX

```
grasp_exportcsv(g, 'nodes.csv', 'edges.csv');
grasp_exportcsv_signal(g, x_1, 'heat.csv');
grasp_exportcsv_signal(g, x_2, 'wn.csv');
```

Figure 6: *Matlab code to export a graph and two graph signals.*

```
...
\usepackage{tikzgraph}
...
\begin{document}
...
\begin{tikzpicture}
  \flatgraph[colormap=jet,showcolormap=true,%
    minvalue=-0.35,maxvalue=0.15,nodesize=13pt]%
    {nodes.csv}{edges.csv}{heat.csv}
\end{tikzpicture}
...
\end{document}
```

Figure 7: *LaTeXcode for Figure 3.*

Remark 1. The package `tikzgraph` also implements the command `\backgroundflatgraph` allowing to have a background image similarly to the Matlab output of Figure 5, and `\stemgraph` to plot the signal using stems. In addition, the package `tikzmatrix` implements the command `\tikzmatrix` to plot a matrix.

Interfacing with External Toolboxes

```
list(cur_dep).url          = ['https://github.com/epfl-lts2/gspbox/'...
                              'releases/download/0.6.0/gspbox-0.6.0.zip'];
list(cur_dep).name         = 'gspbox/';
list(cur_dep).root_dir     = 'gspbox/';
list(cur_dep).init_script  = 'gsp_install';
list(cur_dep).start_script = 'gsp_start';
list(cur_dep).optional     = 1;
list(cur_dep).ref_bib      = 'https://arxiv.org/abs/1408.5781';
```

Figure 8: *Excerpt of `grasp_dependencies_list` showing how the GSPbox [1] is included in GraSP.*

```
% Make the GSPbox available to Matlab
grasp_start_opt_3rd_party('gspbox');

% Convert the GraSP structure g to a GSPbox structure and back
g_gsp = grasp_to_gspbox(g);
g     = grasp_from_gspbox(g_gsp);
```

Figure 9: *Start the GSPbox and convert a graph back and forth between GraSP and GSPbox.*

Raw list of Functionalities

- Graph Structure
- **Classical graphs**
- Adjacency from distance matrix
- Laplacian matrices
- **GFT matrix computation**
- Perform the GFT on signals and operators
- **CSV import / export**
- **LaTeX commands to plot graphs, graph signals, and matrices**
- **Graph Plotting with Graph Signals**
- **Graphical User Interfaces**
- External toolboxes installed and loaded automatically

Main Differences with the GSPbox [1]

- GSPbox: graph cluster plotting
- GraSP: interface with GraphVIZ
- Optimized for GUIs
- Edge weights using colors VS thickness
- Latex commands

Graphical User Interfaces

```
% WN and a low-pass filter
x = normrnd(0, 1, grasp_nb_nodes(g), 1);
H = grasp_fourier_inverse(g, diag(1 ./ (1 + g.eigvals)));

% First GUI
grasp_animate_iterated_operator_gui(g, H, x);

% Second GUI
grasp_gft_gui(g, x);
```

Figure 10: *Example code to use the GUIs of Figures 11 and 12.*

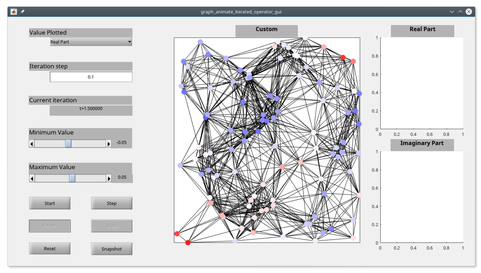

Figure 11: *A Graphical User Interface optimized to iterate an operator (here a low pass filter) on a signal (here a WN) and animate this iteration.*

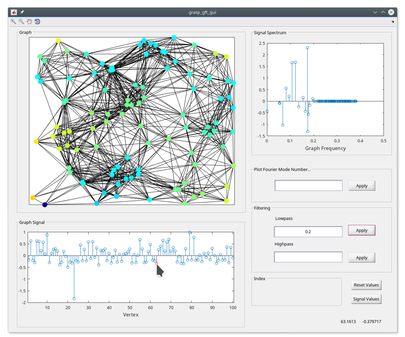

Figure 12: *Another GUI displaying a graph signal on a graph (top left), on the vertex axis (bottom left), and the associated GFT (top right). The last two figures can be used to interactively change the graph signal, or its GFT. Ideal low-pass and high-pass filters are available (bottom right).*

Conclusion and Future Work

- Matlab toolbox implementing many useful functions to generate, manipulate and display graphs and graph signals.
- Efficient implementation of external (optional) dependencies without including the code in the archive.
- Future work: Use the Matlab graph object instead of a custom graph structure, and backport this object definition for older version of Matlab and GNU Octave (WIP).
- Future work: Merge the GSPbox and GraSP.

References

[1] N. Perraudin, J. Paratte, D. Shuman, V. Kalofolias, P. Vandergheynst, and D. K. Hammond. GSPBOX: A toolbox for signal processing on graphs, Aug. 2014.

ICASSP 2017

Signals and Graphs processed and displayed using the GraSP toolbox (http://gforge.inria.fr/projects/grasp or https://github.com/STAC-USC/GraSP). This work was supported in part by NSF under grants CCF-1410009, CCF-1527874, CCF-1029373, and Labex Milyon.